**Topic 7 CLC-Pipeline-with-Static-vs-Dynamic-Scheduling**

**Problems in this exercise refer to the following sequence of instructions:**

| Instructions | Execution Cycles |
| --- | --- |
| FP\_Add/Sub | 2 |
| FP\_Multiply | 3 |
| FP\_Divide | 5 |
| INT\_Divide | 4 |

1. **INT\_add R3, R1, R2**
2. **Br\_Untaken offset, R3**
3. **fp\_add F6, F1, F2**
4. **fp\_div F6, F2, F3**
5. **fp\_sub F2, F3, F6**
6. **fp\_mult F6, F2, F3**
7. **fp\_add F5, F2, F1**
8. **fp\_sub F1, F5, F3**

**For this problem, assume:**

| IF | ID | EX | MEM | WB |
| --- | --- | --- | --- | --- |
| 200ps | 120ps | 150ps | 190ps | 100ps |

**William Stallings Link for Simulating the Programs:** [**CPU Cycles Program Run Simulation**](http://www.ecs.umass.edu/ece/koren/architecture/windlx/main.html)

1. **Indicate All Dependencies that can be found in this instruction set, and their Type (RAW, WAR, RAR, WAW, Structure, Control)**

| **Dependency Type** | **Register in Conflict** | **Instr. Nos. Involved** |
| --- | --- | --- |
| **RAW** | **R3** | **0, 1** |
| **WAW** | **F6** | **2, 3** |
| **RAW** | **F6** | **2, 4** |
| **RAW** | **F6** | **3, 4** |
| **WAR** | **F2** | **3, 4** |
| **WAW** | **F6** | **3, 5** |
| **Control** | **N/A** | **3, 5** |
| **RAW** | **F2** | **4, 5** |
| **Control** | **N/A** | **3, 6** |
| **RAW** | **F2** | **4, 6** |
| **Control** | **N/A** | **5, 6** |
| **RAW** | **F5** | **6, 7** |
| **Structural** | **N/A** | **6, 7** |

1. **Assume “no forwarding” in this pipelined processor and apply (Inst. 1, 2, and 3), from above. Indicate hazards and add NOP (Stall) instructions to eliminate them.**

| **Cycles Where**  **Hazard Eliminated** | **Instruction No. Delayed** | **Pipeline Phase Delayed** | **Reason Delay Occurred** |
| --- | --- | --- | --- |
| **5-9** | **2** | **MEM, WB** | **WAW dependency for register F6** |

1. **What is the total execution time of this instruction sequence in the “no forwarding” pipelined processor?**

| **Total Execution Time for “No Forwarding” Pipeline Processor** |
| --- |
| **4630 ps** |

1. **Now, assume a “forwarding” pipelined processor and apply (Inst. 1, 2, and 3), from above. Indicate hazards found, and add NOP (Stall) instructions to eliminate them.**

| **Cycle No. Where**  **Hazard Eliminated** | **Instruction No. &**  **Pipeline Name Delayed** | **Reason for Change** |
| --- | --- | --- |
| **5-9** | **2, EX** | **WAW hazard in register F6** |

1. **What is the Total execution of this instruction sequence in the “forwarding” pipelined processor?**

| **Total Execution Time for “Forwarding” Pipeline Processor** |
| --- |
| **3510 ps** |

1. **What is the Speedup due to Using a Forwarding Pipelined Processor?**

| **Processing Speedup due to Forwarding** |
| --- |
| **4630 ps / 3510 ps = 1.319** |

**Results Discussion**

1. **Dependencies**
   1. **RAW in R3 in instructions 0 and 1**
   2. **WAW in F6 in instructions 2, 3**
   3. **RAW in F6 in instructions 2, 4**
   4. **RAW in F6 in instructions 3, 4**
   5. **WAR in F2 in instructions 3, 4**
   6. **WAW in F6 in instructions 3, 5**
   7. **Control dependency in instructions 3, 5**
   8. **RAW in F2 in instructions 4, 5**
   9. **Control dependency in instructions 3, 6**
   10. **RAW in F2 in instructions 4, 6**
   11. **Control dependency in instructions 5, 6**
   12. **RAW in F5 in instructions 6, 7**
   13. **Structural dependency in instructions 6, 7**
2. **No forwarding hazards**
   1. **Cycles 5-9, instruction 2 is delayed, MEM and WB phases are delayed, reason for delay is WAW dependency in register F6**
3. **No forwarding execution time**
   1. **4630 ps**
4. **Forwarding hazards**
   1. **Cycles 5-9, instruction 2 is delayed, EX phase is delayed, reason for delay is WAW dependency in register F6**
5. **Forwarding execution time**
   1. **3510 ps**
6. **Speedup**
   1. **4630 ps / 3510 ps = 1.319**

**Conclusion**

**In conclusion, this exercise was an addition in learning data dependencies on execution efficiency and the delays caused by dependency hazards with and without forwarding. The streamlined flow achieved with data forwarding eliminates the need for stall instructions and results in an 8.9% speed increase in execution time.**